Unity经验积累

# 1.1素材导入

在finder中批量选中想要导入的文件夹，并复制粘贴到unity中，注意：assets和packages是根目录下的分开的两个文件夹

在切割素材时，根据作者绘制的像素比例进行切割，不同素材的锚点不同

# 1.2瓦片绘制

Window->2D->Tile Palette中有unity内置的场景绘制工具tile palette

把切割后的图片素材导入创建的新调色盘中

在sample scene->2D object->Tilemap中选中需要的网格地图，作为绘制的目标

在palette中选中瓦片，便可以绘制

# 1.3实现叠层

在renderer->additional settings->sorting layer中创建新的叠层，遮挡原理：后创建的叠层会遮挡新创建的叠层

创建叠层后要在sorting layer中设置物体所处的叠层

## 1.3.1如何实现连接的物体有不同的叠层？

在grid中创建不同的tilemap，并设置不同的sorting layer

A screenshot of a computer

Description automatically generated

设置了不同的sorting layer后，根据需要再设置order in layer，其中数值越大，显示的优先级就越高

Scene->tilemap focus允许高亮显示现在正在绘制的图层，在多layers的情况下比较好用

A video game screen with a white character

Description automatically generated

# 1.4有规则和动态瓦片

在Tiles中另外创建新的文件夹Rule Tiles，根据需要，2D->Tiles->Rule Tile创建规则瓦片，先设置default sprite

要使多个瓦片随机出现，Rule Tile inspector->Output选择random

编辑好规则瓦片后，将其放在想要使用的tile palette中，使用合适的工具绘制

在编辑规则瓦片时，根据瓦片顺序的排列可以设置优先级（放在前面的优先级更高）

在Tiles中另外创建新的文件夹Animated Tiles，根据需要，2D->Tiles->Animated Tile创建规则瓦片，同理

Inspector中设置minimum speed，此为图片的切换速度

# 1.5设置人物及基本组件

Rigidbody 2D：使该物体成为刚体，会收到重力的影响

Collider 2D:使该物体具有碰撞体积

### 1.5.1TileMap collider和composite collider的结合用法

为瓦片地图添加这两个组件后，在tilemapcollider中勾选used by composite,使检测碰撞时，将绘制的瓦片地图视为一个整体，而不是每一个单元格分别检测

要使tilemap静止，不掉落，在刚体组件中选择static

为了避免2D游戏中人物在z轴上的旋转，在刚体组件中冻结z轴rotation

要使人物在爬墙时持续检测，collision detection设置为continuous

# 1.6创建及配置新输入系统

Assets->Scripts文件夹用于保存所有的代码

Edit->Project settings->Input Manager是旧输入系统不容易进行跨平台操作

Edit->Project Settings->Player->Other Settings->Configuration->Api Compatibility Level设置为.NET Framework，允许利用更多C#特性来完成任务

Edit->Project Settings->Player->Other Settings->Configuration->Active Input Handling 设置为Input System Package(New)

Window->Package Manager 安装Input System Package

Settings文件夹中创建子文件夹Input System,创建Input Actions(Input Action Asset),可命名为InputControls

A screenshot of a computer

Description automatically generated

左侧Action Maps 是输入集，用于切换输入系统，比如游玩时和控制UI时

在新创建的Action Maps中创建Actions, Action Type在控制移动时需要捕获和处理连续的输入数据，所以设置为Value, Control Type 定义了与 Action Type 配合的具体输入控制类型，即什么样的输入设备或控制器用于生成该输入数据。在控制移动时应该选择2D Vector

左上角添加Control Schemes用于控制切换输入系统，后在Actions中勾选

以上是手动创建输入系统的例子，还有自动创建输入系统的方法

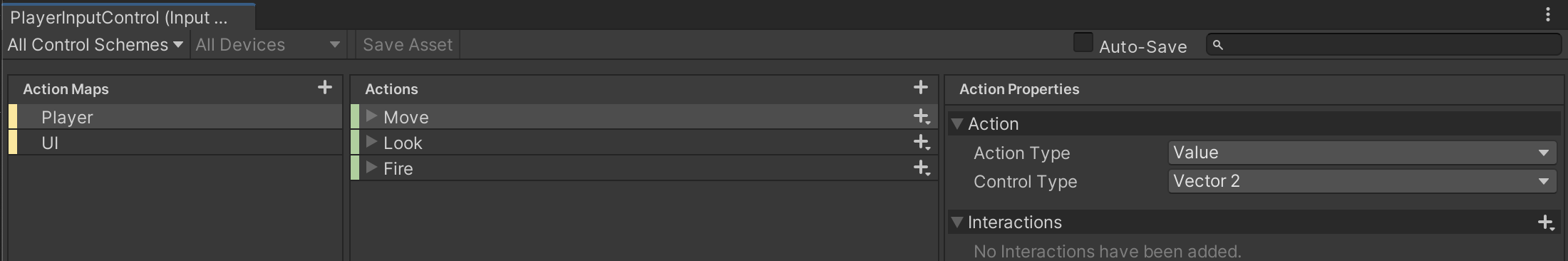
A screenshot of a computer

Description automatically generated

### 1.6.1自动创建输入系统

为要配置输入系统的物体添加组件Player Input, Create Actions，在指定的文件夹中保存

直接生成了输入系统：



选中要生成c#类的控制系统文件，在inspector中勾选generate c# class,生成脚本

在PlayerController中调用UnityEngine.InputSystem,并将之前的控制系统类实例化

# 1.7实现移动和反转

一旦一个物品被挂上了rigidbody2D组件后，它的坐标就由该组件驱动

实现反转的方法：1.在transform中将x轴的值变成负值

2. 在sprite renderer中，勾选Flip x或y

# 1.8实现人物跳跃

在输入系统文件中添加新的action,命名为jump，类型为button

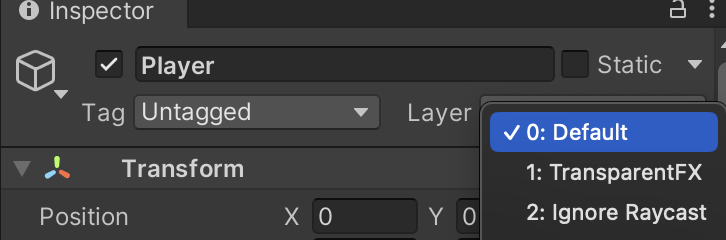
修改gravity scale和jump force，测试合适的手感

# 1.9物理环境检测及绘制

连续跳跃的问题：不希望人物持续不断的跳跃

为了实现物理检测，（敌人或玩家），都创建通用脚本

layerMask的碰撞涂层和inspector右上角的物理检测碰撞图层是一致的：



为想要检测碰撞的物体添加指定的碰撞图层，在inspector窗口中设置便可

#### 1.9.1如何在scene中绘制出脚底的碰撞检测范围？

调用drawWireSphere()

#### 1.9.2如何解决人物跳起来贴着墙会静止不动？

在Settings中创建文件夹Physics Material,在文件夹中创建新的2D材质normal

# 1.11创建人物基本动画（帧动画）

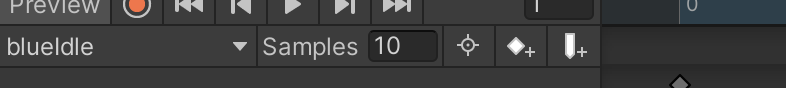
为想要创建动画的物体添加animator组件,在assets中创建新的文件夹animations,player

Window->Animation->Animator中打开动画控制器

#### 1.11.1如何使用animator

先选中Player, Window->Animation->Animation窗口打开

在animation中先添加图片，再设置采样率



#### 1.11.2在animator中设置动画的切换条件

在animator中，在两个state之间创建一个transition

在parameter中，添加要作为动画切换条件的变量A black and white text

Description automatically generated with medium confidence

创建新的脚本PlayerAnimation用于管理控制所有动画,在代码中绑定animator中的变量，把需要的数值同步过去，再在animator中设置切换的条件

# 1.12人物行走设定

#### 1.12.1如何实现某个按键被按下时强制人物进入走路状态

先确定动画切换的条件，是速度达到某个值，在输入系统中创建新的button binding，在playerControl中使用回调函数，通过按键实现速度的控制

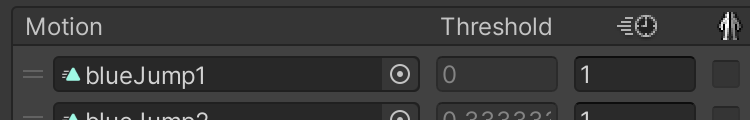
# 1.13创建人物跳跃动画

#### 1.13.1动画混合树的使用

在unity中，混合树是一种用来平滑混合多个动画的机制，它的blend type决定了

动画的混合模式，如何通过一个或多个参数进行混合

在animator中右键，创建混合树，进入混合树后在右上角设置blend type

添加motion field 

在base layer中设置切换的条件，根据需要

在脚本中通过SetBool()传入参数

#### 1.13.2在人物落地时立刻奔跑，landing动画还未结束，怎么解决？

在base layer中，让player能够立刻从landing动画的播放状态退出，设置transition到exit

# 1.14实现人物下蹲逻辑和动画

在animator中创建子状态机，通过将相关的动画归纳到一个子状态机中，简化动画的切换

#### 1.14.1怎样从下蹲状态退出？

1. Make transition to exit state, set the parameter to (isCrouch == false)
2. Make transition to the base layer

在这里选择第一种方法

完成代码修改，发现人物下蹲时依然可以左右平移

#### 1.14.2人物下蹲时的碰撞体大小修改

在unity中，修改碰撞题体积的时候发现是按照碰撞体中心点进行修改，而不是按照精灵锚点进行修改

在代码部分实现逻辑：下蹲时修改碰撞体大小，并进行相应的位移

取消下蹲时还原之前的碰撞体参数

记录：offset: -0.05, 0.95 size:0.7, 1.9（原始）

Offset:-0.05, 0.8 size:0.7, 1.6 （下蹲时）

#### 1.14.3使人物下蹲时无法移动

在人物移动的功能前加一个isCrouch的判断

# 1.15人物属性及伤害计算

为敌人添加box collider，用于攻击判定

#### 1.15.1取消敌人和玩家的碰撞，防止人物推动敌人或骑在敌人头上

Collider组件中，layer overrides，通过碰撞体的重载实现

在inspector窗口中，设置不同的layer mask，用于碰撞和摄像头渲染，并为不同的物体设置相应的layer，另外，renderer中的sorting layer用于物体的遮挡逻辑

#### 1.15.2实现人物与敌人碰撞时扣血

为敌人添加capsule collider，用于玩家受击判定，将其isTrigger设置为true,将这个椭圆设置为触发器，而不是碰撞器，并剔除enemy layer

#### 1.15.3创建角色基本属性的脚本，并挂载到player和boar上

#### 1.15.4创建攻击脚本attack，同上

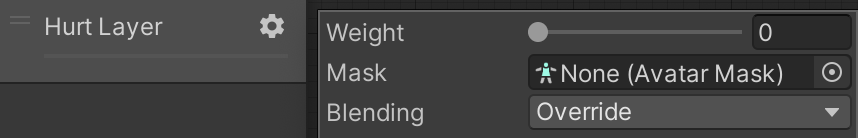
#### 1.15.5如何实现无敌帧

通过无敌总时间，无敌时间计时器，time.deltatime 和 update()函数实现

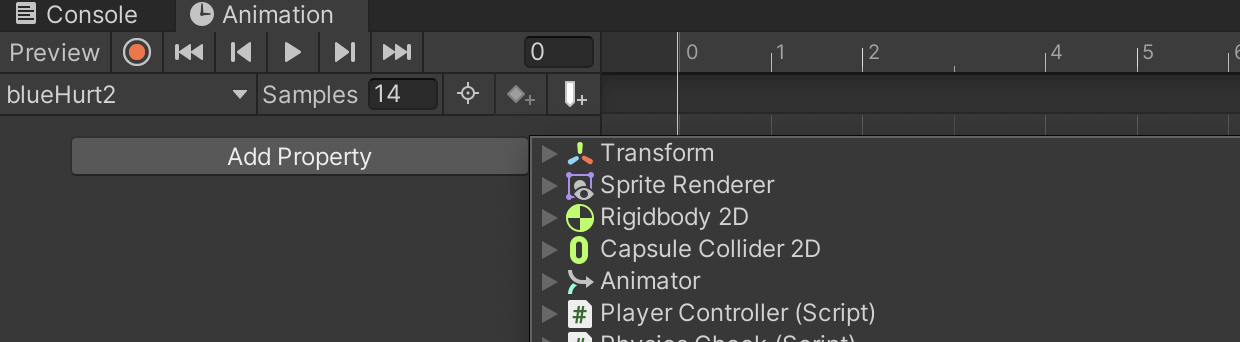
# 1.16受伤和死亡的逻辑和动画

#### 1.16.1实现玩家受伤时闪烁

在base layer外创建新的图层hurt layer（animator里）



Weight是动画播放的权重，blending有override和additive两种模式，additive是叠加层



在animation中，点击add property，可以在动画播放的过程中修改组件的参数（实现动画播放到某一帧或几帧时无敌）

Add Property->Sprite Renderer->Player: Material.\_Color

更改Material.\_Color.a,即rgb中的alpha值，使其在一个完整的采样周期中从1和0.5两个离散的值之间跳转n次（n>=1）,n为闪烁动画的闪烁次数

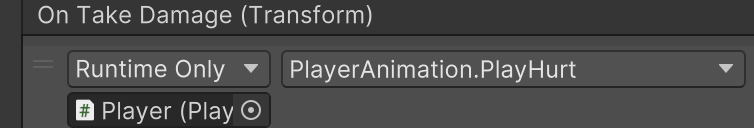
将blueHurt2(闪烁动画)添加到叠加层中，entry->new empty state->blueHurt2

为它设置新的触发条件trigger:hurt 注意⚠️：trigger参数不能在update函数中设置

要使用unity的事件，在文件开头include 引擎的命名空间UnityEngine.Events

#### 1.16.2实现挂载组件的个体，在受伤时朝着受伤的反方向被击退

需要先知道attacker的方向（左侧或右侧），再向反方向位移



事件触发时，调用PlayerAnimation的函数方法PlayHurt()

#### 1.16.3如何启动事件

UnityEvent<B> a;

a?.Invoke(b);

将向反方向位移的函数和启动受击动画trigger的函数（两个）注册到事件中

#### 1.16.4解决动画播放完毕后因为isHurt还是true，无法控制角色

在受击动画中添加behaviour，OnExit函数中将isHurt设置为false

综合之前的内容，使玩家死亡时播放死亡动画，如何使死亡动画仅播放一次？选中动画，在inspector中将loop time取消勾选，这样就能使动画单词播放

注意：仅有连接到any state的动画可以循环播放

# 1.17三段攻击动画的实现

创建新的attack layer，权重调整为1

类似hurt layer，先创建new empty state，作为进入attack state的上级状态

动画播放逻辑：在上一层攻击动画还未播放完之前，有攻击输入，则播放下一段攻击

Animator中创建参数bool isAttack和trigger attack

在attack1: exit time中设置0.9，当动画播放90%之前，都可以进入attack2的动画播放，如果动画完全播放，则退出连击判定

在inputSystem中，添加攻击按钮

PlayerController中，编辑攻击代码

#### 1.17.1连击计数器变动太快，导致玩家无法迅速进入下一个攻击阶段

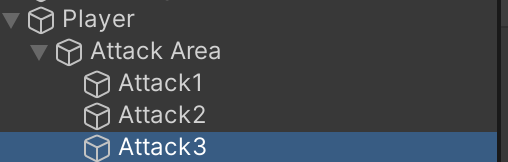
解决办法：不使用combo作为连击计数器

在animator中，attack1添加新的表现AttackFinish

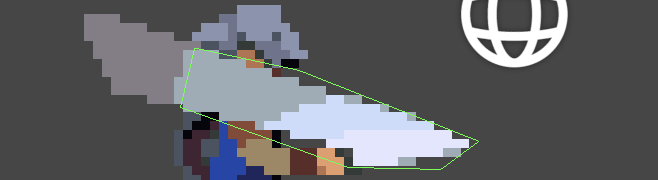
# 1.18实装攻击判定

观察动画，在攻击动画的哪几帧需要攻击判定

通过trigger触发，但希望player身上不要有太多碰撞体，在player上添加一个子集物体



为每个攻击范围添加碰撞体组件Polygon Collider 2D，设置为Trigger

希望在动画blueAttack1的第四帧播放

原理：在动画中添加关键帧，让动画播放到第四帧的时候启动Attackn

为Attackn挂载Attack脚本，设置damage为10

因为Attack脚本中使用的OnTrigger方法没有特别指明对象的类型（Player/Enemy）导致玩家本身的碰撞体在执行攻击后也会触发该方法，导致扣血

解决方法: Polygon Collider 2D->Layer Overrides->Contact Capture Layer勾选Enemy

#### 1.18.1测试时发现攻击不扣血

原因是boar身上有两个碰撞体：用于攻击玩家的box collider和用于受伤判定的capsule collider，需要调整判定的优先级，Capsule Collider 2D->Layer Overrides->Layer Overrides Priority设置为10或比0高的任何数值

为了是一次攻击只让野猪扣血一次，在Character组件中设置invulnerable time为1

#### 1.18.2问题1:玩家在攻击时依然可以继续移动

在PlayerController中的FixedUpdate中添加新的条件

将之前创建的光滑材质改名为wall，新创建有摩擦力的材质normal

摩擦力系数设置为0.4，将player的材质改为normal

如果需要在每一段攻击之前加一个小突进，在Animator->AttackFinish behaviour中将OnStateEnter函数注释，否则玩家在攻击时会停留在原地

#### 1.18.3问题2：玩家在原地持续不断的攻击时不会有伤害

出现这个现象是因为OnTriggerStay2D有个休眠时间，它被触发一段时间后自动停止。要想再次触发该函数需要稍微移动其中一个碰撞盒，也就是玩家需要移动到不同位置攻击野猪它才会掉血。后面的课程会让野猪动起来，这样这个问题就不存在了。当然也可以修改OnTriggerStay2D的休眠时间，使之变长，这样即使在原地攻击野猪它也会持续掉血。修改OnTriggerStay2D的休眠时间方法：点击Edit -> Project Settings，然后选择Physics 2D，如下图所示，在General Settings中将Time To Sleep修改为一个较大的值，如10000（默认情况下是0.5）

# 1.19野猪-基本的移动逻辑和动画

创建父类Enemy和子类Boar

移动逻辑：持续不断的走到撞墙为止，然后回头

出现问题：因为没有删除子类中自动生成的代码，导致Update()函数覆写了父类Enemy的Update()函数

希望野猪在发现正前方有玩家，或者被玩家从背后攻击时，切换到run；过了一定的丢失时间后，切换回walk巡逻状态

注意：在地图墙壁绘制的时候，默认情况下碰撞体会按照图片的像素内容进行绘制，导致墙壁不是平的，修改方法是选中地图图片，打开 **Sprite Editor** 在左上角选择 **Custom Physics Shape** 来进行修改。将不平的墙壁物理形状改成如图的**白色方块**

# 1.20野猪-撞墙判定和等候计时

为野猪挂载physicsCheck，设置offset为-1.3，使其可以检测面前有没有地面

在physicsCheck中修改，使其可以进行碰墙检测

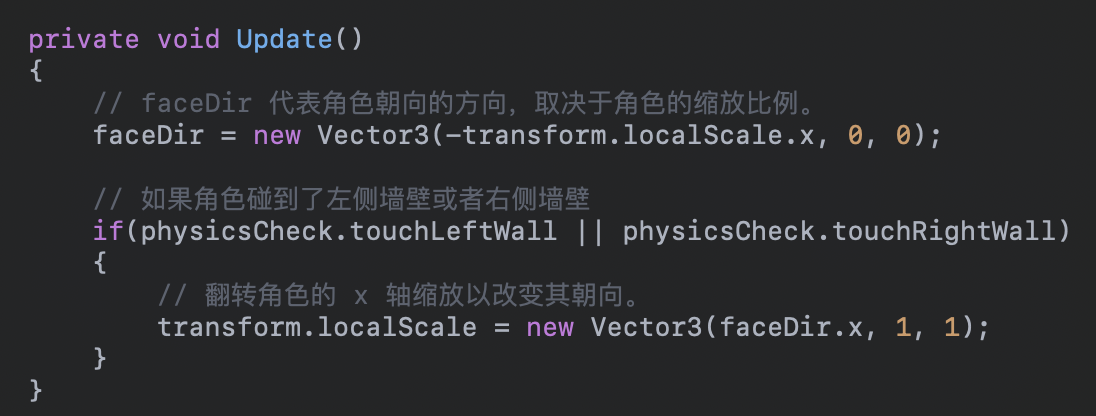
在boar的physcisCheck脚本中修改leftOffset和rightOffset

#### 1.20.1实现自动调整left,rightOffset，使其可以自动定位到碰撞体的左和右侧

Platform->Composite Collider 2D->Geometry Type更改为polygons，使墙体的判定范围不仅只有一条轮廓

#### 1.20.2实现野猪左右巡逻

问题：野猪撞墙时，会持续翻转

原因：

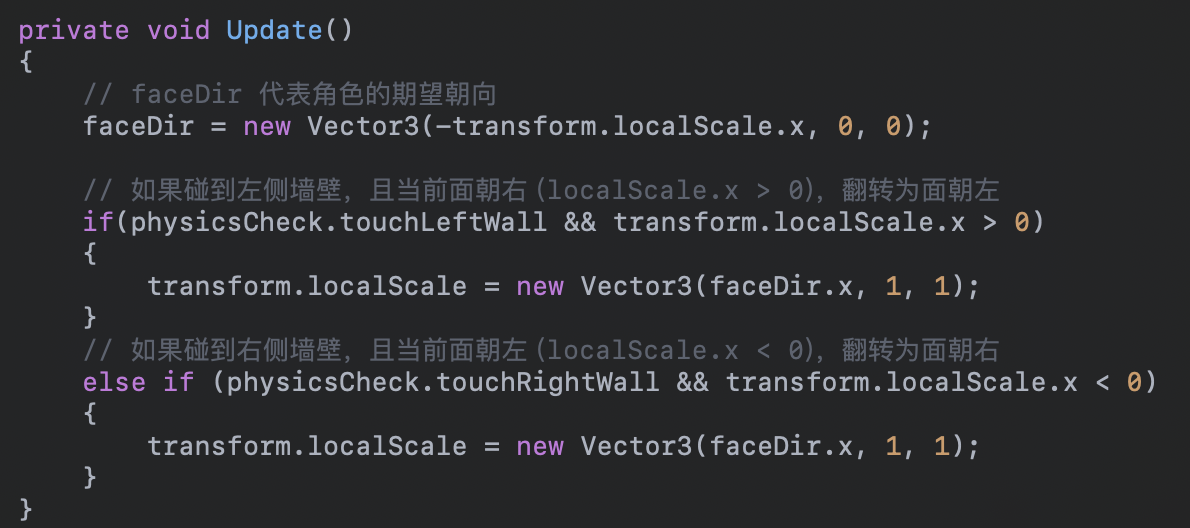
角色向左移动碰到左墙壁时，当前帧的faceDir为(-1, 0, 0)

If语句的判断条件为真，于是localScale改变为(-1, 1, 1)，即执行了翻转

在下一帧，faceDir更新为(1, 0, 0)，即移动方向向右，而if语句的条件应该为false，就不会出现反复翻转的现象，为什么？

原因未知

解决办法：在翻转之前判断角色的朝向，只有朝向错误的时候才进行翻转

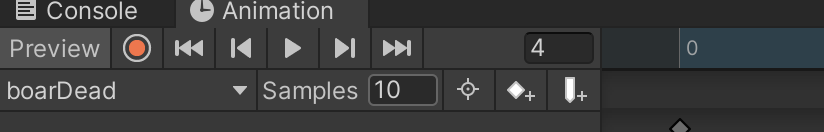


#### 1.20.3实现撞墙后等待倒计时结束再进行翻转

#### 1.20.4实现在撞墙等候时间时，播放idle动画

# 1.21野猪-受伤和死亡的逻辑和动画

在动画中添加函数事件



右边的白色竖线，允许在指定的关键帧，调用函数

# 1.22有限状态机&抽象类多态

抽象类中，只有函数的生命，没有实现

在Enemy文件夹中创建基类BaseState，编辑抽象类

创建野猪巡逻状态BoarPatrolState

在状态机内试图获取敌人的参数，需要将特定的组件改为public，如何在inspector中隐藏显示？

通过添加attribute[HideInInspector]

#### 1.22.1地图上有坑的时候，野猪本应该停下，但滑动着往前走了

Enemy中，Move()函数的实现前要加上非等待时作为条件

# 1.23追击状态的转换

Physics2D.BoxCast()的概念：*BoxCast* 在概念上类似于沿特定方向在场景中拖动一个框。 可以检测和报告任何与盒子接触的物体。

返回值：此函数返回一个 [RaycastHit2D](https://docs.unity3d.com/ScriptReference/RaycastHit2D.html) 对象，该对象引用了被长方体击中的碰撞体（如果未击中任何内容，则结果的 Collider 属性将为 NULL）。*layerMask* 可用于仅在某些图层上选择性地检测对象（例如，这允许您将检测仅应用于敌方角色）。

不知道该函数的返回值时这么写：



光标移动到var上可以看到

在Enemy中写好切换的方法

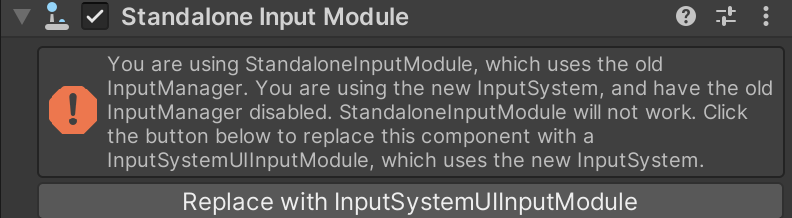
在Scripts文件夹中创建Utilities文件夹，用于存储工具类

按照逻辑，进入chase状态后更改野猪的速度

# 1.24创建人物状态栏

Hierarchy->create UI->Canvas会自动顺便生成EventSystem

Replace with inputsystemuiinputmodule





Actions Asset切换到自己创建的Player Input Control

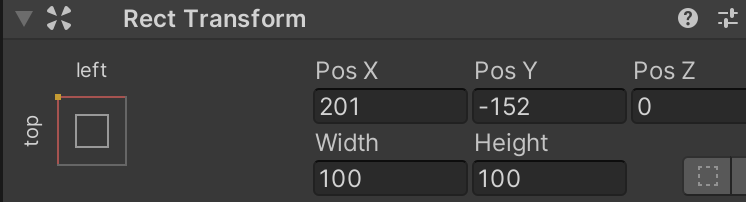
Canvas->Canvas Scaler->UI Scale Mode->调整为Scale with screen size，作用是根据实际游戏的画面分辨率进行布局

Reference resolution调整为游戏测试时的画面分辨率

Match改为0.5，综合显示器长高

Reference Pixel Unit，设置为16，图片素材的像素

在gameObject Canvas->create UI->Image



定位到左上角，按住option+shift+left click，完成定位，拖拽后看到大概位置

Health Fill Green更改填充方式为Horizontal

全选Health有关的GameObject, create empty parent: Health Bar

在切割人物图片时，使用mask组件，并取消勾选Show mask Graphic

# 1.25血量更新逻辑的实现

原理：在填充时，调用fill amount变量



在Scripts文件夹中创建UI文件夹，创建脚本PlayerStatBar，用于人物属性UI控制

如果想使用代码中UI相关的组件，需要先调用命名空间UnityEngine.UI

#### 1.25.1如何实现跨物体或场景传输数值？

ScriptableObject：一种持久化能存储在项目当中的资产性的文件类型，以.asset为后缀名

创建CharacterEventSO，继承ScriptableObject



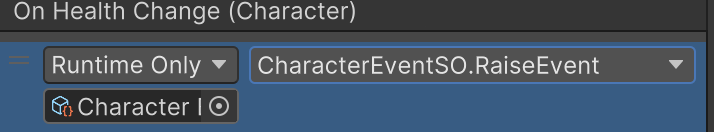
创建资产文件，文件名为CharacterEventSO，可以在project窗口中的加号显示并添加

使用UnityAction委托，方便订阅与激活

在Asset文件夹中创建新的文件夹Events,创建CharacterEventSO

#### 1.25.2如何订阅和传递

在玩家的Character脚本中创建事件OnHealthChange<Character>，使用该事件进行广播



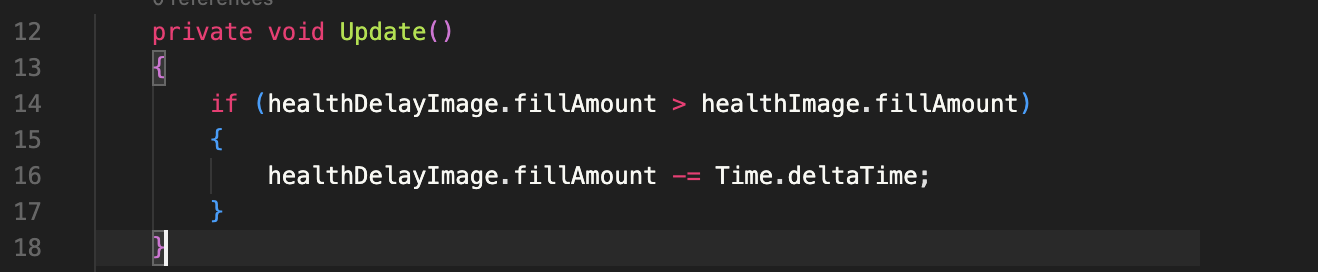
配置该事件

创建新的Game Object：UI Manager用于管理所有的UI行为，并进行监听

#### 1.25.3关于注册事件

在OnEnable()中注册事件，在OnDisable()中取消注册，可以有多个代码订阅同一事件

#### 1.25.4实现人物扣血血条渐变



在update中实现，在时间的修正值上可以乘以自定义的速度变量，修改速度

# 1.26摄像机跟随及攻击抖动的实现

Package Manager->注册工具Cinemachine

添加新的GameObject->Cinemachine->2D Camera，在Follow和Look at中传入player的transform组件

CinemachineVirtualCamera->Body->Tracked Object OffsetY设置高一些，默认是以人物的脚底锚点为基准点跟随

调整Dead Zone Area，为摄像机设置缓冲值，人物在Dead Zone中移动时，摄像机不会移动

#### 1.26.1为了不让摄像机穿帮，需要添加边界限制

在Extensions中选择添加Cinemachine Pixel Perfect:在像素旋转或产生畸变的时候不会产生扭曲，保持单位像素

添加confiner 2D：限定一定的区域，来控制摄像机的移动范围，但需要传入Collider 2D类型的Bounding shape type作为参数

创建新的空GameObject，Bounds,设置为Trigger,并传递给Confiner 2D

#### 1.26.2转换场景时，获得当前场景的边框位置

创建新Tag：Bounds，赋给Bounds物体

需求：CinemachineVirtualCamera获得Confiner组件，并通过变量值查找Bounds物体，并将它的collider自动添加到Bounding shape type

解决办法：CameraControl脚本

#### 1.26.3实现震动反馈

VirtualCamera添加ImpuseListener组件，用于监听震动，需要震动源CinemachineImpulseSource监听震动的事件

2D游戏勾选：Use 2D Distance

创建新的空GameObject : CameraShake,为其挂在脚本ImpulseSource, Impulse Shape选择震动类型，下来菜单显示震动波幅类型

#### 1.26.4通过代码，在每次攻击时启动震动效果

创建新的ScriptableObject-VoidEventSO，创建新的SO文件CameraShake Event

# 1.27音源设置和音效播放

创建Audio Manager，并挂载Audio Source脚本，将想要播放的音频添加到Audio Clip

勾选Play on Awake，在运行游戏的时候自动播放

想要循环播放，勾选Loop

添加另外一个Audio Source，用于播放音效，前一个用来播放bgm

为了使用AudioMixer，在代码中调用UnityEngine.Audio;

以攻击为例子：Attackn每次启动的时候都需要播放音效，可以把要播放的音效直接挂载到它们上面，用它们去通知Audio Manager，接下来要播放什么音效

先在Attack上标记用哪个clip，再用事件的方式进行广播

每一个场景中创建一个空GameObject：BGM，只挂载Audio Definition

#### 1.27.1Audio Mixer混音台的使用

Window->Audio Mixer用于将音频混合进行输出

Settings文件夹创建Audio Mixer文件夹，创建新的资产文件Audio Mixer：Main Mixer

为Master轨道创建两个子轨道FX\BGM，打开Audio Manager的两个Source组件分别进行配置

补充：在Main Camera的组件CinemachineBrain中把Update Method从Smart update更改为Fixed Update，可以消除不必要的抖动

# 1.28水和荆棘的逻辑实现

为创建的水体添加Tag：设置为Water

创建新的Spike瓦片地图，设置sorting layer，并使用TileMap Collider2D，和Composite Collider将多个荆棘的碰撞体合一,将它的Body Type 改为Static

为Spike挂载Attack代码

# 2.1场景转换-人物可互动标识

为了调整每一个场景中的结构，删除新场景中的相机

## 2.1.1hierarchy中多开场景

拖拽新场景到hierarchy，并关闭其他场景的grid, 所有和地图，敌人有关的内容，都保存在新的场景中，老场景中只保存UI, Manager, Virtual Camera

在Scenes中创建新场景Persistant, 把老场景中保存的东西移动到persistent

## 2.1.2创建可互动宝箱

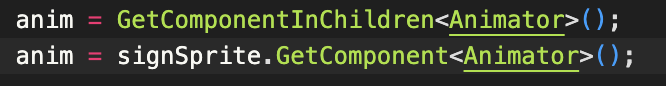
为场景中的宝箱添加碰撞体，设置为trigger，即可发生交互行为的范围

创建按钮，设置为player的子物体，并为其挂载Animator脚本

平时Sign物体处于Disable状态，改名为Sign Sprite，创建父物体Sign，挂载Box Collider 2D，修改位置和大小，使其处于Player的脚下，设置为Trigger

在Player下创建子物体时，默认的layermask也是Player，修改为Default，将Attack也一并修改

因为Animator挂载在子物体Sign Sprite上，所以需要用函数方法获取子物体上的Animator



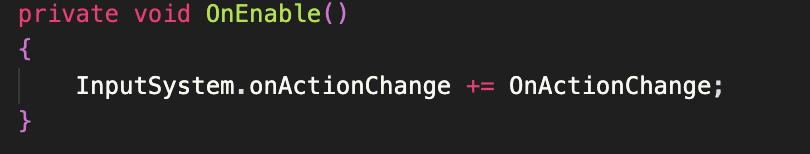
有两种写法，但在游戏开始时，Sign Sprite是关闭的，所以第一种写法没用，使用第二种写法

### 2.1.2.1人物翻转时，交互标志的方向翻转

让标识对应Player的localScale进行翻转

## 2.1.3根据输入系统播放不同动画

为输入系统变化事件，注册函数OnActionChange



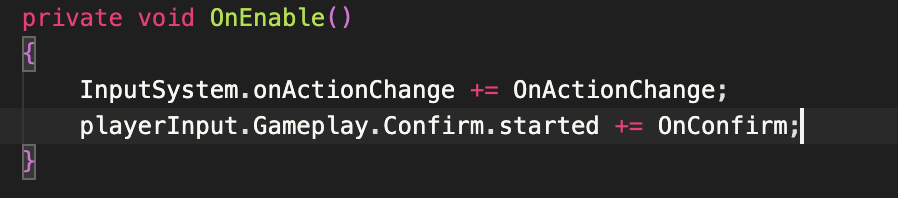
# 2.2场景互动的逻辑实现

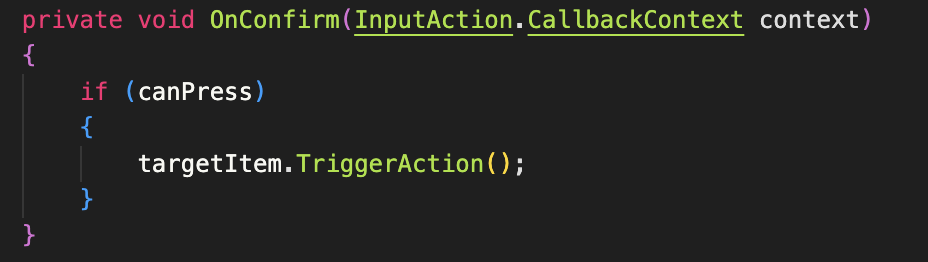
在Utilities创建新的接口脚本IInteractable

逻辑：如果canPress，切按下了指定的按键后，就可以互动

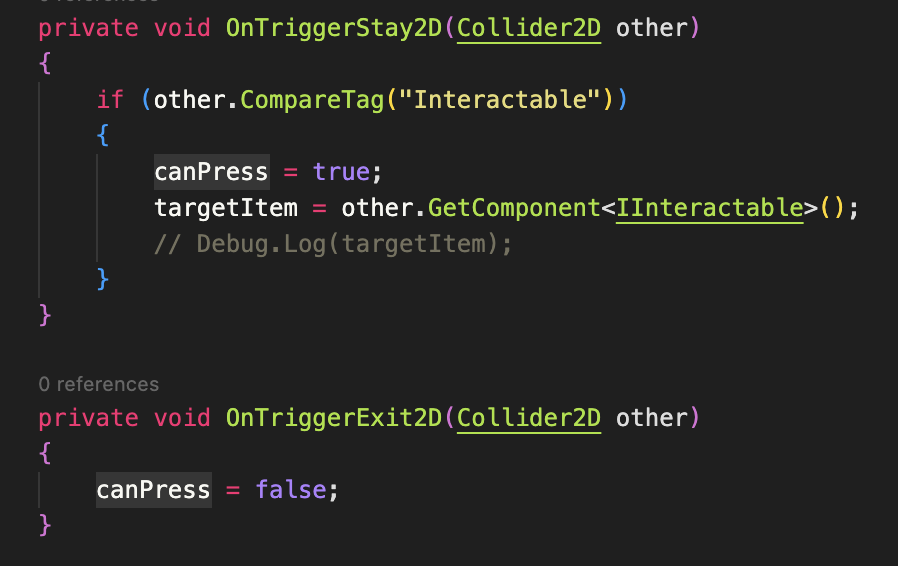
在PlayerInputControl中自定义一个新的按键Confirm，类型是button

原理：Player下有子物体Sign, Sign有子物体Sign Sprite，Sign Sprite的Sprite Renderer是关闭的，但上述的物体都是激活的，在Chest本身的脚本中实现了IInteractable接口，在Sign的脚本中注册了事件





每当playerInput.Gameplay.Confirm被激活时，也就是E键被按下时，如果canpress为真，则调用Chest脚本中的函数。



每当Sign本身的碰撞体，待在实现IInteractable接口的游戏物体的Trigger范围内，则canPress为true，退出Trigger范围时，canPress为false

## 2.2.1为打开宝箱添加音效

为Sign 挂载脚本Audio Definition,并设置SO和音源

## 2.2.2 使用传送门切换场景

原理：unload老场景，load新场景，并调整player的位置，所以需要保存一些变量，比如切换到哪个场景，以及玩家的坐标

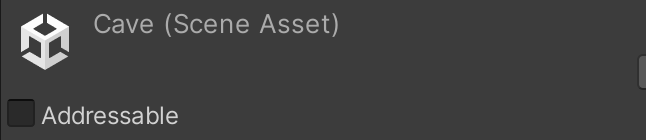
在TeleportPoint中保存Position to go，但当前场景被卸载的时候，传送门组件也会变disable，所以要在Persistent场景中保存,创建SceneLoader进行全局场景切换，所以要用到SO

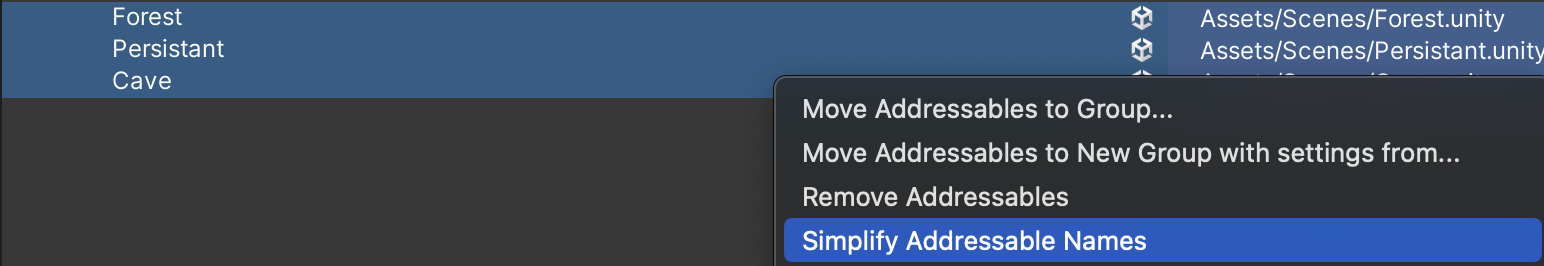
# 2.3场景管理和切换

注册Addressable（可寻址资源加载程序）工具到My Assets

Windows->Asset Management->Addressables->Group

完成配置后，选择任何场景，都可以将其添加到Addressable中的Group中



在Addressable中简化命名

为了频繁跨场景调用GameObject，比如敌人，需要先将其变成Prefab(预制体)，在Asset文件夹下创建子集文件夹Prefabs, 直接将要作为预制体的object拖拽进来，就可以使其成为预制体，之后仅需要对单个预制体进行修改，就可以更改全部的实例

将敌人添加到Addressable中,为了知道每次要加载什么场景，添加新SO

为了区分加载的是场景还是菜单，添加新的Enums：Location, Menu

## 2.3.1Addressable的使用

调用命名空间using UnityEngine.AddressableAssets后，可以通过AssestReference直接访问添加进Addressable工具的游戏资源的引用

# 2.4 场景加载后的执行逻辑

加载新场景后，需要重新给currentLoadedScene赋值

进行场景转换时，需要将人物关闭：开始场景转换时关闭，结束场景转换时开启人物，这样就不会因为地形的消失而下落

## 2.4.1 传送后重新加载摄像机边界

添加事件AfterSceneLoadedEvent，使用广播和监听的方法进行重新加载

但是在修改完后，发现第一次加载场景时，得不到边界

解决办法：在SceneLoader中不在Awake()中进行异步加载场景，而在Start()中使用LoadNewScene加载场景，添加相应的变量

# 2.5场景淡入淡出效果

在Persistent场景中创建新的fade canvas

Canvas组件中的sorting order，设置的比main canvas的Canvas组件更高，比如10

Fade Canvas 创建子物体 UI->image 起名Fade Image, rect transform组件中shift+option点击右下角，使其填充满整个屏幕，在颜色中修改Alpha值，就可以实现淡入淡出

取消勾选Raycast Target，让其拥有遮挡限制

在scripts->UI中创建新脚本FadeCanvas

在商城添加DoTween到assest中

创建新的SO文件FadeEventSO

# 2.6 主场景制作

设置playerStatBar默认关闭，只有在开始游戏时才开启

选择移动图片工具时，按住v键，可以使其对应相应的点进行自动吸附

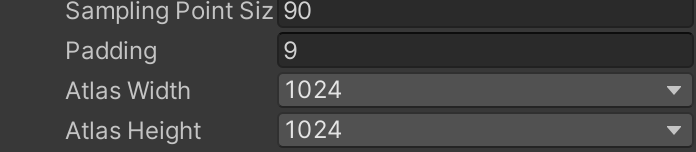
在制作完成的主场景中添加Canvas，sorting order设置为5，代表它会覆盖原本场景中的UI面板

+->UI->Text – Text Mesh pro

## 2.6.1在textmeshpro中使用中文字体

选中下载好的字体，选择create textmeshpro font asset





由于渲染机制问题，可以选择增大画布面积提升文字渲染数量

勾选Color gradient启动启动字体颜色渐变

添加组件Vertical Layout Group以管理子物体的垂直等距排序

## 2.6.2创建按钮

1. 在Canvas中->UI->Button-TextMeshPro创建ui中的button，实现开始游戏，加载等功能



这会创建一个按钮，下面包含一个textmeshpro文本游戏物体

2. 或者直接在canvas中创建text-textmeshpro物体，并挂在button组件

## 2.6.3 实现开始游戏

Data SO->Game Scene创建新的gamescene so文件，重命名为menu

在scene load manager中，把first load scene改成menu，发现运行游戏后人物消失了，原因是firs load position有误差，更改成想要人物在menu中所在位置的坐标

需要：在场景转换时使人物不能移动

在menu中人物不能移动

# 2.7 实现”New Game”逻辑

## 2.7.1 new game时等场景加载完成后再显示血条

记得在挂载character脚本的object身上选中new game event，不然会获取不了血条

## 2.7.2 menu键盘选择选项

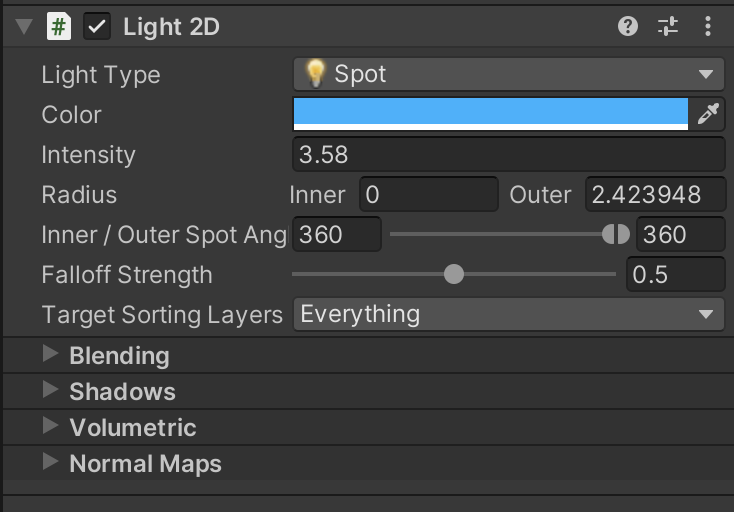
创建代码，将button的值传递给persistent中的event system的first selected

# 2.8 存储点及画面效果

## 2.8.1画面后处理及灯光

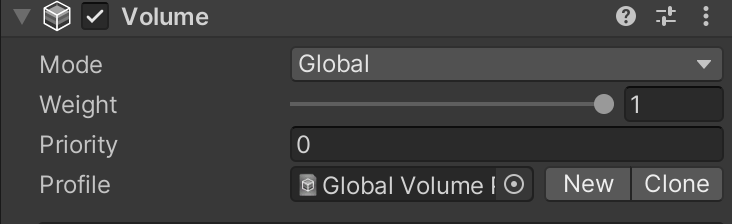


保存后存储点上的符号变蓝，需要增加周围的蓝色光照效果

为符号增加2D光照，设置如下：

## 2.8.2 通过渲染管线进行后处理

创建global volume 

用new创建一个profile

# 2.9数据结构及保存后加载

## 2.9.1单例模式

**单例模式** 是一种 **设计模式**，它确保**一个类在整个程序生命周期内只有一个实例**，并提供一个全局访问点来获取该实例。

**✅ 单例模式的特点**

1. **只有一个实例** 👉 在程序运行过程中，单例类的实例不会被创建多个。

2. **全局访问点** 👉 其他代码可以通过一个固定的方式访问这个唯一实例。

3. **防止外部创建实例** 👉 通过私有构造函数，阻止外部代码创建新对象。

单例模式的应用场景：全局配置、音效管理、缓存管理

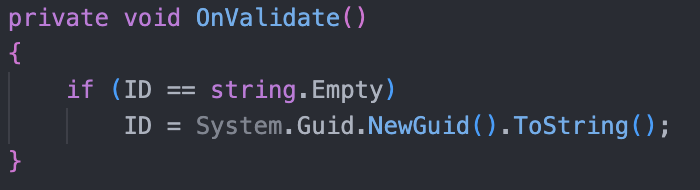
## 2.9.2 用字典存储数据

Data不需要挂载在游戏物体上，所以不需要继承monobehaviour

用Dictionary<string, Vector3>存储角色的位置坐标

用unity自带的GUID(Global Unique Identifier)实现作为key value

## 2.9.3数据定义



OnValidate() 是 Unity MonoBehaviour 中的一个生命周期方法，在 **脚本的 Inspector 面板发生更改时** 自动调用

问题：在游戏开启后立即关闭，不做任何操作，GUID会改变，因为OnValidate方法会检测到inspector面板发生了更改

方法：创建枚举变量